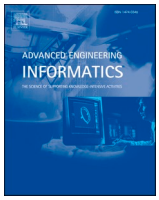
Advanced Engineering Informatics 53 (2022) 101710

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei



Full length article

Digital twin-driven deep reinforcement learning for adaptive task allocation in robotic construction

Dongmin Lee [a], SangHyun Lee [b,*], Neda Masoud [c], M.S. Krishnan [d], Victor C. Li [e]

[a] School of Architecture & Building Science, Chung-Ang University, Seoul 06974, Republic of Korea
[b] Tishman Construction Management Program, Dept. of Civil and Environmental Engineering, Univ. of Michigan, 2350 Hayward St., G.G Brown Bldg., Ann Arbor, MI 48109, USA
[c] Dept. of Civil and Environmental Engineering, Univ. of Michigan, 2350 Hayward St., G.G Brown Bldg., Ann Arbor, MI 48109, USA
[d] Computer Information Systems, Technology and Operations at the Ross School of Business, Univ. of Michigan, Tappan St., 701, Ann Arbor, MI 48109, USA
[e] Dept. of Civil and Environmental Engineering, Univ. of Michigan, 2350 Hayward St., G.G Brown Bldg., Ann Arbor, MI 48109, USA

ARTICLE INFO

Keywords:
Digital Twin
Proximal Policy Optimization (PPO)
Deep Reinforcement Learning (DRL)
Autonomous Robot
Adaptive Task Allocation

ABSTRACT

In order to accomplish diverse tasks successfully in a dynamic (i.e., changing over time) construction environment, robots should be able to prioritize assigned tasks to optimize their performance in a given state. Recently, a deep reinforcement learning (DRL) approach has shown potential for addressing such adaptive task allocation. It remains unanswered, however, whether or not DRL can address adaptive task allocation problems in dynamic robotic construction environments. In this paper, we developed and tested a digital twin-driven DRL learning method to explore the potential of DRL for adaptive task allocation in robotic construction environments. Specifically, the digital twin synthesizes sensory data from physical assets and is used to simulate a variety of dynamic robotic construction site conditions within which a DRL agent can interact. As a result, the agent can learn an adaptive task allocation strategy that increases project performance. We tested this method with a case project in which a virtual robotic construction project (i.e., interlocking concrete bricks are delivered and assembled by robots) was digitally twinned for DRL training and testing. Results indicated that the DRL model's task allocation approach reduced construction time by 36% in three dynamic testing environments when compared to a rule-based imperative model. The proposed DRL learning method promises to be an effective tool for adaptive task allocation in dynamic robotic construction environments. Such an adaptive task allocation method can help construction robots cope with uncertainties and can ultimately improve construction project performance by efficiently prioritizing assigned tasks.

1. Introduction

Construction is known for its physical and psychological stress on workers as well as for its hazardous working conditions. In addition, the construction industry suffers from a lack of skilled laborers [1], an aging worforce [2], and stagnant labor productivity growth compared to other industries (e.g., manufacturing) [3]. Meanwhile, recent robotics technology advancements are addressing safety concerns and stagnant labor productivity issues in construction work: on-site autonomous robots have been used to help human workers and/or reduce repetitive tasks. However, to accomplish more complicated tasks and to reap more benefits from the autonomous robot in a dynamic construction site environment, robotic construction technology should move beyond mere autonomy and be truly adaptive to site conditions. Therefore, there is a need for a technical approach to teach robots an adaptive strategy (policy). The adaptive policy refers to a task allocation policy — the robot's way of prioritizing assigned tasks and executing them in given states. For example, if the robot aims to reduce project cost, its policy will prioritize tasks that can reduce project cost and execute them accordingly.

In recent years, several researchers have demonstrated the potential of deep reinforcement learning (DRL) —which enables an agent to learn an action policy through repetitive interactions with a virtual simulation environment— for a more adaptive agent policy [4,5]. Such adaptivity may make robots better suited to interacting in more uncertain environments without specific prior knowledge or instructions (e.g., task

* Corresponding author.
E-mail addresses: dmlee@cau.ac.kr (D. Lee), shdpm@umich.edu (S. Lee), nmasoud@umich.edu (N. Masoud), mskrish@umich.edu (M.S. Krishnan), vcli@umich.edu (V.C. Li).

https://doi.org/10.1016/j.aei.2022.101710
Received 17 November 2020; Received in revised form 11 January 2022; Accepted 28 July 2022
Available online 12 August 2022

allocation rules). Once the DRL agent has learned the adaptive policy in the simulation environment, this trained policy can be deployed in real robotic applications. However, even though DRL may adapt well to static and/or deterministic environments [4], from a practical point of view, it is not thoroughly researched whether a DRL agent can learn an adaptive task allocation policy in an unstructured construction environment where the robot might encounter and deal with changing site conditions over time.

In order to address this gap in knowledge, we have developed and tested a digital twin-driven DRL framework to investigate DRL's potential for adaptive task allocation in a dynamic and unstructured robotic construction environment. The digital twin is a digital representation of an asset or system and mimics its real-world behavior with the aim of managing, planning, predicting, and demonstrating current and future construction processes and models via synchronized representations [6,7]. The digital twin can capture overall changing site conditions in near real-time and creates a virtual simulation environment with which a DRL agent can interact. For our purposes here, "real-time" means an event or function is processed instantaneously (i.e., there is no noticeable delay between the action and its effect). In this environment, the DRL agent receives positive or negative rewards through the interactions and accordingly updates the parameters of the policy network. Therefore, this paper presents a new paradigm of the adaptive task allocation method in robotic construction by investigating the potential of the DRL algorithm. We tested whether the DRL agent consistently learns task allocation policy in the digital twin environment with a case project where virtual robots load and assemble prefabricated concrete bricks to build a mock-up bridge. Then, we validated whether such a policy shows higher performance than a rule-based imperative algorithm when applied in different environment that reflect changing site conditions.

The main contribution of this study is to demonstrate the potential of a digital twin-driven DRL framework for finding a task allocation policy that is adaptive to dynamic robotic construction environments. Such an adaptive task allocation method can help construction robots cope with more complicated and uncertain tasks than previously available methods, ultimately improving project performance.

2. Digital twin and reinforcement learning in construction

2.1. Unstructured and evolving construction environments

There are many different tasks in construction project such as bricklaying, painting, brick assembly, material loading, and bulldozing. Some of them are too repetitive and hazardous to human workers. Robotics can be a promising solution to address the issues. Robots can take over repetitive or physically demanding tasks or they can assist human workers to perform tasks safely and efficiently. However, construction robots (i.e., robots operating in the construction site) should perform tasks in unstructured and evolving site conditions [8]. For example, humans and equipment are moving around without predetermined path, delivery rate of raw materials from the factory to the site is changing over time according to the transportation condition, and the operation patterns of heavy equipment can be different every day. For this reason, construction robots should understand changing site environment quickly, and make decisions (e.g., task allocation) adaptively considering the surrounding environment.

2.2. Digital twin

The concept of a "digital twin" as a data centric asset management and control has emerged over the past decade mainly in manufacturing and production domains [9]. Although there is no commonly agreed definition of the term "digital twin" in the context of the construction domain [10], this concept should be used as either digital representations of physical assets (e.g., building objects, materials, robots, and workers) or systems mirroring a project's real-world behavior [11]. Particularly, in this paper, we emphasize the enabling potential of a digital twin for testing different "what-if" scenarios (i.e., to anticipate the effect of a process given a predefined situation) in robotic construction. Compared to a normal simulation imitating, real-world process or system, an ideal digital twin enables a more data-driven, real-world-oriented simulation based on cyber-physical synchronicity. The digital twin can be updated or evolve with an updated data feed from physical space, which may reduce discrepancies between physical and virtual spaces. Thus, simulation with a digital twin can reflect reality better than the normal simulation (i.e., a reduced discrepancy between reality and simulation) and any results from a digital twin simulation can be directly actuated to address real-world problems. Furthermore, various synthetic conditions can be simulated by using a digital twin model and historical dataset. For example, based on a historical dataset of raw materials' arrival rates at a jobsite, the arrival rate of raw material can be defined as a model parameter to simulate different arrival rate conditions that are likely to happen in the real-world construction site. As another example, after collecting historical traffic patterns of equipment at a construction site, we can extrapolate different traffic patterns that are likely to occur in the future. Thus, the digital twin is central to our study because it can be used to simulate a variety of robotic construction conditions.

Additionally, an IoT sensor designed to collect physical assets' conditions, convert them into digital signals, and connect to the internet for data communication [12]. can be a desirable method for enabling cyber-physical synchronicity. With the aid of IoT sensors, physical assets' conditions and their digital twin counterparts can be synchronized in near real-time. In addition, the Robot Operating System (ROS), an open-source robotics development framework that can be used to control the motion of real robots, allows real robots and their corresponding twins to be synchronized in terms of their motion [13]. Thus, we may assume that the digital twin can mirror all entities based on IoT sensors and ROS that exist in robotic construction.

2.3. Reinforcement learning

Usually, the RL is one of three basic machine learning methods, alongside supervised learning and unsupervised learning [14]. RL differs from the other two methods, which learn from massive datasets that are collected once and then reused, in that RL uses a trial-and-error feedback loop that requires active interaction with an environment to collect data [15]. An RL agent receives positive/negative rewards for its actions from the environment, with the aim of learning to select actions that maximize the accumulated reward [15]. This approach is particularly advantageous for robot control problems in construction wherein it is expensive, time-consuming to collect a large dataset for training a model [16]. As long as there is a realistic construction simulation environment for the RL agent to interact with, RL can learn control policy without the collected dataset [17].

2.4. Markov decision process

The typical RL algorithm uses the formal framework of the Markov Decision Process (MDP) which determines the action given a full set of observation data [18]. MDP is built on an assumption that future process states depend only on the current state. We assume that our task allocation problem will follow the MDP process. MDP is defined by the tuple $M = (S, A, P, R, \gamma, T)$, where S is the set of states of the system (e.g., locations of robots, assembly progress of robots, and availability of the raw materials); A is the set of actions (e.g., assembly sequence and delivery speed); P defines the probability for a transition from the current state to a next state; R defines the reward according to the selected agent's action; γ is a discount factor, ranging from 0 to 1, a reward R that occurs t steps in the future from the current state, is multiplied by γ^t to emphasize its importance to the current state; and T is the length of the

learning iterations. At each decision point t, the agent observes the current state $s_t \in S$ and chooses an action $a_t \in A$ then, gets a new state s_{t+1} with the probability P and receives a reward $r_t \in R$. The objective of an RL is to find a policy π_θ, parameterized by θ, with the objective of maximizing the cumulative reward ($\mathbb{E}_\tau$) throughout the episode:

$$\max_\theta \mathbb{E}_\tau \left[\sum_{t=0}^{T} \gamma^t r_t(s_t, a_t) \right] \quad (1)$$

where τ denotes the trajectory sequence.: $\{s_0, a_0, r_0, s_1, a_1, r_1, \cdots, s_T, a_T, r_T\}$

2.5. Policy gradient algorithm

The policy gradient (PG) algorithm is a type of RL technique that relies on optimizing parameterized function θ to optimize the policy directly with respect to the long-term cumulative reward [15]. The PG algorithm has higher learning efficiency than other RL methods when the state and action space are large (high-dimensional) or continuous, making it suitable for controlling robot movement and/or operations in a complicated environment [19]. The PG computes an estimate of the gradient of the long-term cumulative reward expected of a given policy, and then updates the policy parameters in the gradient direction. If we use the cumulative reward as an objective function, we can derive the following equation for estimating the gradient [18]:

$$\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta log \pi_\theta(a_t|s_t)\hat{A}_t] \quad (2)$$

where π_θ (a_t is action and, s_t is a state at time t) is a stochastic policy (i.e., probability of taking an action a_t at given state s_t), $\hat{A}_t$ is an estimator of the "advantage function" at time t [21]. The advantage function describes an expected average reward when following the policy over the sampled trajectories [20].

2.6. Proximal policy optimization

While various forms of the PG method exist, proximal policy optimization (PPO) has demonstrated comparable or better performance than recent PG approaches and is much simpler to implement [20]. PPO enables multiple updates per minibatch sample to promote sample efficiency and guarantees the stability of policy optimization by limiting the update amplitude of the policy [22]. The PPO algorithm defines a "clipped surrogate objective" which constrains the size of the policy change at each learning step using a clip:

$$L^{CLIP}(\theta) = E_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t] \quad (3)$$

where ϵ is a clip hyperparameter, and $r_t(\theta)$ is the probability ratio defined in Equation (4):

$$r_t(\theta) = \frac{\pi_{\theta(a_t|s_t)}}{\pi_{\theta_{old}(a_t|s_t)}} \quad (4)$$

This ratio describes the difference between our new and old policies, which enables the clip function to discourage large policy changes. This not only leads to less variance in agent training in a construction environment which includes dynamic and stochastic processes but also ensures that the agent does not make irrecoverable errors while training.

3. Deep reinforcement learning for adaptive task allocation in robotic construction

3.1. Autonomous robots in construction

Globally, the construction sector is considered a less digitized and automated industry than others and its labor productivity has stagnated accordingly [3]. Robotic automation, whereby autonomy is achieved partially or fully using robots, has been considered as a solution to address the challenge above [23]. Automation and robotics types in construction can be categorized into four general groups: (1) off-site prefabricated systems, (2) on-site automated and robotic systems, (3) drones and autonomous vehicles, and (4) exoskeletons [23]. In this paper, we focused on the second group, "on-site automated and robotic systems." Within this category are various definitions for the robot's levels of autonomy (LoA), which, according to the Society of Automotive Engineers (SAE), range from 0 (fully non-autonomous) to 5 (fully autonomous). Our research is focused on LoA 4 wherein a robot can autonomously perceive their surrounding environments and adaptively allocate tasks under pre-defined site conditions [24]. When this level of autonomy is reached, a robot can perform diverse construction tasks in unstructured and evolving environments. Our ultimate goal for future work is to achieve LoA 5 wherein the robot can operate fully autonomously beyond human-level performance even in a completely new environment.

Bock et al published series of handbooks in construction robotics which focuses on the implementation of automation and robot technology to renew the construction industry. They academically organized and compiled knowledge in the fields of 1) robot-oriented design and management [24], 2) robotic industrialization [25], 3) construction robots [26], 4) site automation [27], and 5) ambient integrated robotics [28]. Gharbia et al. reviewed papers on autonomous robot technology for on-site building construction [29]. In their review, they noted that autonomous robotics has recently become the most cited research area in robotic construction and, in particular, that this technology has been widely adopted in areas such as additive manufacturing [30], robotic assembly [8], automated concrete spraying [31], distributed robotic construction [32], and more. They concluded that autonomous robots help to reduce many manual processes, and therefore could address low productivity issues in construction. Melenbrink et al also reviewed the research trends of autonomous robots in construction in terms of three groups of tasks: site preparation (earthmoving and leveling), substructure (anchoring and foundations), and superstructure (load-bearing elements, façade, plumbing, and wiring) [23]. They have highlighted the autonomous robot's potential for helping with repetitive and dangerous tasks, which also helps to solve the shortage of skilled construction labor. However, they also noted limitations on current robots for achieving enough autonomy (i.e., levels 4 of LoA) in an unstructured environment. In other words, the current autonomous robot still requires human supervision when applied in an unstructured environment like construction. On the other hand, Goessens et al [33] and Latteur et al [34] suggested another autonomous robot concept —using a drone-based automated masonry construction method to build small-scale structures made out of the concrete precast element— and Petersen et al [35] suggested and tested the use of multiple, collective, and construction robots to build a small-scale structure made out of building blocks. In addition, Iturralde et al recently suggested a cable driven parallel robot for the installation of real-world curtain wall modules [36]. Their applications showed the high feasibility of using autonomous robots to build real-scale structures without human supervision, but these demonstrations were also limited to well-structured environments.

In a practical demonstration of adaptive robotics, Komatsu [37] and Volvo [38] developed an autonomous dump truck robot for material handlings in a construction site. Their robots uses a global positioning system, light detection and ranging, radio detection and ranging, and multiple sensors (e.g., load sensor and inertial measurement unit) to observe the environment and react accordingly when performing material handling tasks. Similarly, Brayman construction developed a rail gantry crane robot for automated rebar work [39], and INTSITE developed an autonomous tower crane robot for automated material handling [40]. They combined multiple sensors, computer vision, machine learning to enable data-driven machine controls. In contrast, in the control of such an adaptive robot, the spiking neuron algorithm [41], fuzzy logic [42], and heuristic algorithms [43–45] have been applied in

previous studies. However, until now such robots have operated only in preprogrammed construction scenarios, and thus they may not yet be reliable enough to be used on a dynamic construction site. In short, current autonomous robots demonstrate highly promising potential for enhancing project performance in well-structured repetitive task and/or simply programmed construction scenarios. However, to accomplish more complicated tasks and to achieve more benefits from autonomous robots in a dynamic construction site, robots should be adaptive to the environment.

3.2. Potential of DRL for adaptive task allocation in construction

Deep reinforcement learning (DRL) is an application of deep neural networks which can be used to better approximate functions in RL for more accurate and consistent learning convergence [46]. The deep neural network can be used to approximate any of the following components of RL: value function $\widehat{v}(s;w)$, policy $\pi(a|s;w)$, and model (i.e., state transition function and reward function) [14]. The value function is a function of states and/or actions that represent how good each state, or state-action pair, is. From data about the agent's state and the environment, the policy dictates actions for the agent to take. The state transition function takes the current state and returns to the next state in response to an event. The reward function determines rewards for actions. Specifically, the function uses parameters w, as a weight in deep neural networks, and its learning consists of finding the right w by iteratively adjusting those w along gradients that promise less error [15]. The deep neural networks enable RL to create better representations (e.g., features), even when the state space or action space are too large (i.e., high-dimensional) to be completely known [47]. DRL has achieved success in solving difficult problems (e.g., autonomous vehicles) which were very challenging when approached using classical RL methods [48–50]. In particular, DRL has proven effective for addressing robot control problems such as locomotion [51], grasping [52], manipulation [53], and imitation [54]. Moreover, other researchers have noted the potential DRL holds for task allocation problem in a shop floor environment [55]. As the success of these other applications indicates, the DRL algorithm may be a good alternative for an adaptive task allocation method in robotic construction.

On the other hand, while a DRL model with well-designed rewards can adapt well to a certain static environment, even a well-trained DRL model may not adapt to a dynamic environment because such a model may be overfitted to a specific environment [4]. In general, a construction project consists of different, complex processes which involve temporary alliances with various stakeholders [56,57], some of which can introduce conditions that the agent interprets as stochastic. For example, from the agent's point of view, movement patterns of workers in a construction site that follow a certain probability distribution or randomness can be perceived as stochastic because the agent cannot precisely predict their patterns. In an RL domain, if there are stochastic conditions in an environment—which fluctuate randomly, regardless of the agent's actions, the agent cannot uniquely determine the environment's next state [58–60]. For this reason, an agent's actions must be tried many times in order to form a reliable estimate of its expected reward [15]. Learning in stochastic conditions has proven one of the most difficult RL problems because the trained policies do not necessarily converge to a single solution [61–63]. For these reasons, it has not been thoroughly determined whether a DRL agent can find an adaptive task allocation policy, and whether such a policy gives us reliable control performance in a robotic construction environment which includes both dynamic and stochastic processes. There is thus a growing need for simulating such an environment and testing the potential of DRL for more adaptive task allocation methods in robotic construction.

4. Digital twin-driven deep reinforcement learning for adaptive task allocation in robotic construction

The objective of the study is to fill the aforementioned gap in the research by developing and testing a digital twin-driven DRL framework used to investigate DRL's potential for adaptive task allocation in a robotic construction environment with dynamic and stochastic processes at play.

4.1. Framework overview

The digital twin-driven DRL framework consists of three components as shown in Fig. 1. This framework shows how project data is collected from the construction site and updated in the digital twin. The framework also explains how the DRL agent interacts with the digital twin to observe a new state and receive rewards for their actions. At its core, this framework utilizes the digital twin as a dynamic learning environment that observes overall site states in real-time. Such overall state observation is necessary for the DRL agent to learn an adaptive task allocation policy, which in turn increases performance at the project level rather than at an individual robot's level. For example, the agent may learn a policy that allocates tasks by comprehensively considering the state of other robots, workers, and heavy equipment. Moreover, this type of learning framework is new because the policy learned from the digital twin environment has a reduced simulation-to-reality gap (i.e., a discrepancy between simulated results and reality) which enables more realistic (i.e., practically applicable to real-world problem) policy learning.

The IoT sensors attached to physical assets collect real-time sensor data (e.g., locations and properties of materials, and locations of robots) from a construction site (Fig. 1-(a)) and transmit the data to the digital twin (Fig. 1-(b)). Then, "as-is" building information modeling (BIM) can be updated based on "as-designed" BIM and IoT sensor data collected from the site. A DRL agent can interact with the dynamic digital twin environment and test their actions in different 'what-if' scenarios (Fig. 1-(c)). The DRL agent is a centralized single agent and can govern all robots with only one policy network. The agent earns positive or negative rewards for its task allocations (e.g., brick loading and assembly sequence) in response to states observed and reported by the digital twin and updates the parameters of the policy network accordingly. If the agent's behavior improves the project's performance, it receives positive rewards and vice versa. As a result, the agent can find an adaptive task allocation policy that increases the overall project performance by maximizing average rewards per episode in response to the robotic construction environment.

4.2. Deep reinforcement learning via proximal policy optimization

In order to learn adaptive task allocation policy effectively, an appropriate deep reinforcement learning algorithm should be selected. While DRL algorithms fall into a variety of categories, the main approaches have used value-function-based algorithms such as deep Q-learning [64]. However, Q-learning can only be applied to solve problems with discrete action spaces (i.e., only a finite and discrete number of action sets) [22] and it is not widely used to learn stochastic policies, which would be useful in dynamic construction environments [65]. Since construction robots may be operated in either discrete or continuous action space (i.e., infinite and continuous number of action set) or in an environment which needs stochastic policies in the future, we used proximal policy optimization (PPO) in this paper—a state-of-the-art policy gradient-based RL algorithm. PPO is designed for use both in discrete and continuous action spaces [66], and showed better overall performance than other policy gradient algorithms such as A2C and ACER [20]. However, we do not claim that PPO is the only solution for adaptive task allocation or that it performs better than value-function based algorithms.

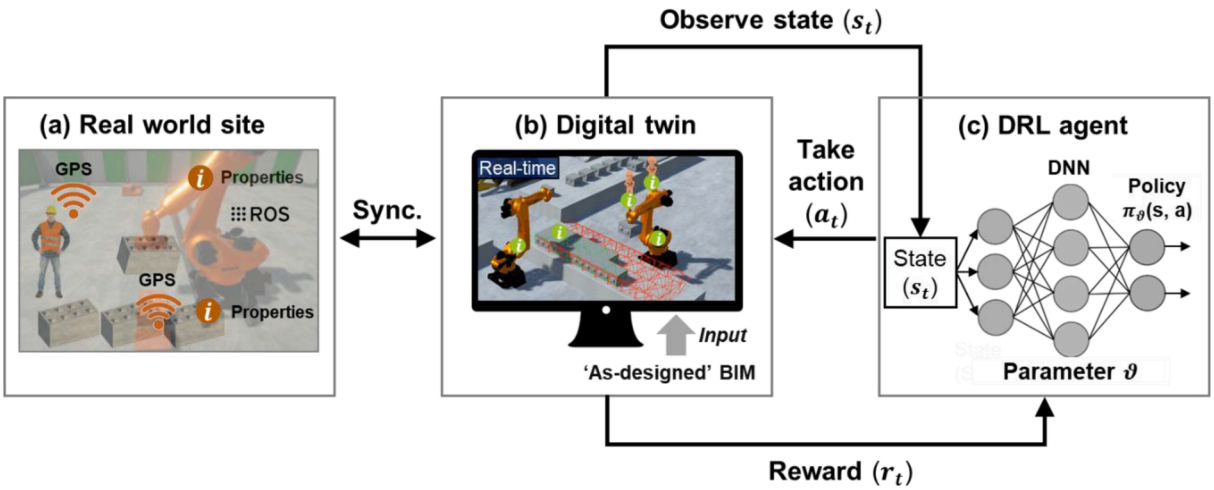


Fig. 1. Digital twin-driven DRL framework.

4.2.1. Deep neural network for PPO

We used the PPO algorithm for policy learning and applied a deep neural network (DNN) for more stable learning in continuous and/or high-dimensional states (e.g., truck location, truck speed, and assembly progress). Furthermore, when RL is applied in future real-world construction sites, features should be extracted directly from raw data. In these cases, DNN will improve feature extraction. A digital twin represents overall site conditions such as material delivery status in a stockyard, locations of workers and equipment, and construction task progress (e.g., how many and which bricks are assembled). Such information (features) can be collected by virtual sensors (e.g., computer vision, ray-casting) (Fig. 2) or manually in the digital twin and preprocessed to be loaded as input state variables into the network.

The DNN model consists of n hidden layers (each layer may consist of i, j units respectively). These variables can be changed according to the complexity of the digital twin environment and/or size of the state and action space of robots (Fig. 3). Rectified linear unit (ReLU) layers are used as the activation layer between each layer to account for nonlinearity. In the final activation stage, the actions of each robot are revealed as output given site states. The number of output units will depend on the number of robots governed by the DNN model. The resultant stochastic policy is passed back to the digital twin, where it probabilistically determines robots' actions and creates new states. Data received by the agent in the form of a positive or negative reward based on their actions is also passed back to the twin, which updates the weight of the network accordingly to reinforce and/or discourage such actions.

4.2.2. Reward functions

In this paper, the goal of our DRL algorithm was to find a task allocation policy adaptive to a dynamic robotic construction environment that includes stochastic processes. The policy aims for robot behaviors which increase overall project performance given site conditions. The policy aims for robot behaviors that increase the overall project performance given site conditions. The performance evaluation metrics in a construction project can include "time," "cost," "safety," "quality," and even "sustainability." Effectively managing these performance metrics throughout the project lifecycle has a direct impact on the success or failure of the entire project [67]. These metrics should be reflected in the reward function so that the DRL agent can increase project performance by maximizing the cumulative reward. In this paper, we used 'time' as a metric of performance because it is one of the most important performance indicators ensuring timely completion of the project [68], and can be easily calculated and converted into a reward. Crucially, both local (short-term) and global (long-term) performance are reflected in the reward function; local performance evaluates from the perspective of a robot, while global performance does so at the overall project level. In general, local performance is evaluated at every iteration according to the agent's behavior, and global performance is evaluated at the end of one learning episode. The reason why both short- and long-term rewards are needed here is to obtain both stable learning and improved project performance in a robotic construction environment. Through short-term rewards, the agent can stably learn the adaptive policy for each task while long-term rewards, which are normally bigger than short-term rewards, ensure that the agent is learning in accordance with the long-

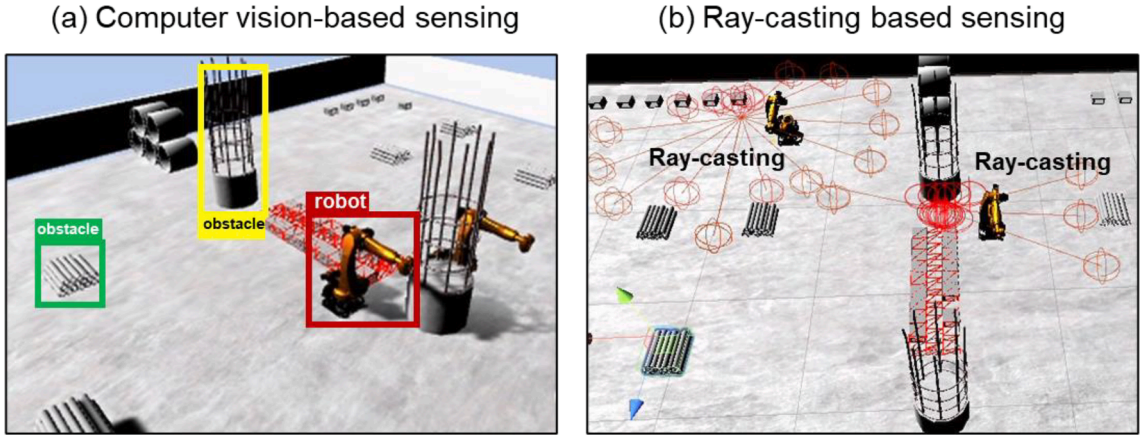


Fig. 2. Information collection based on virtual sensors in the digital twin.

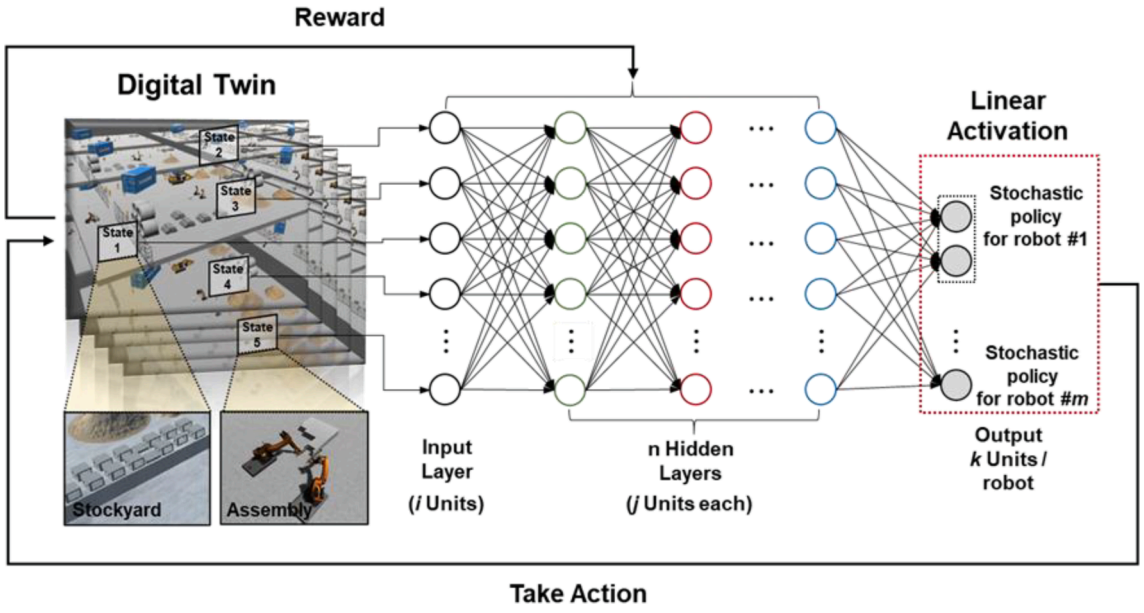


Fig. 3. Deep neural network architecture for robot control learning in digital twin.

term project goal. Accordingly, the DRL agent receives a short-term reward for the agent's performance at each learning iteration (e.g., the time required for a brick assembly) and a long-term reward for the project's performance at the completion of one project episode (e.g., total construction time). For example, if robots have to assemble fifty bricks to complete one bridge, an agent will receive a small short-term reward for each brick assembled and one larger long-term reward when they finish assembling all fifty bricks.

$$r_{total} = \left(\sum r_{short} + r_{long} \right) \tag{5}$$

where r_{local} is a short-term reward, provided for each iteration (action), while r_{long} is a long-term, one time reward that is provided when a learning episode ends.

5. Case Study: Experiments and results

To test the proposed digital twin-driven DRL framework for adaptive task allocation policy learning in a robotic construction environment, we conducted a case study with a hypothetical but realistic project. This case study has three steps: input modeling, DRL agent training, and performance comparison between the DRL and rule-based model. The first step is input modeling, a process that generates hypothetical sensor data in order to create a digital twin environment that includes dynamic and stochastic processes (Fig. 4-(a)). In other words, input modeling replaces the process of collecting data from real-world assets with IoT sensors. Clearly, IoT sensor data ascribed to physical assets are fundamental requirements for the creation of a digital twin, but because such a technique has not yet been implemented, we created hypothetical sensor data that fed into our digital twin. The case project was designed to create different site conditions by changing the simulation model parameters. For example, we can control the average number of workers moving around the site and the assembly speed of robots in the case project. Then the case project is fed into the digital twin so that we can simulate changing site conditions and use it for DRL learning. The model parameters represent dynamic and stochastic processes in robotic construction, using appropriate probability distributions. In the second step, the DRL agent was trained in the digital twin to find an adaptive task allocation policy within the preset parameter ranges, and the stability of DRL policy training was tested by checking the learning curve for two RL indicators (i.e., cumulative reward and episode time) (Fig. 4-(b)). In the last step, we tested the task allocation performance of the trained policy within and outside of the preset parameter ranges. In other words, we created a testing environment that includes both the environment the agent experienced during training and another, unfamiliar, environment to measure the policy performance across both cases (Fig. 4-(c)). Then, we compared the task allocation performance of the DRL model to that of a rule-based model that was designed to take action based on a predefined task allocation rule. For example, we can make a rule to transport and assemble bricks in the order the bricks arrive on a first-come, first-serve basis. Such a rule-based model cannot adapt its behaviors based on the environment because its task allocation behavior remains unchanged throughout the simulation regardless of environmental changes. Therefore, by comparing the task allocation performance of the rule-based model with that of the DRL model, it is possible to evaluate the adaptivity of the DRL model to a robotic construction environment which includes dynamic and stochastic processes.

5.1. Case project

To create dynamic and stochastic site conditions in the digital twin environment, we created a simple but realistic case project in which virtual industrial robots load and assemble prefabricated concrete bricks to construct a small-scale mock-up bridge. The project was designed to mimic the actual dynamic robotic construction process in bridge construction [69–72]. We tried to simplify this case to better understand

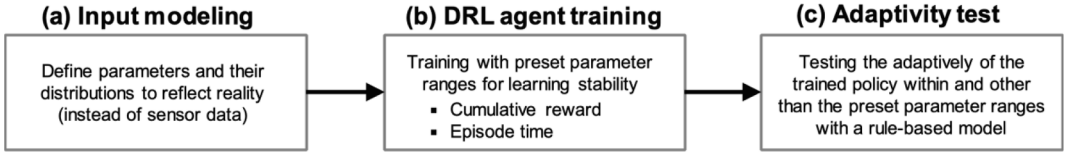


Fig. 4. Overall process of case study.

and explain the agent's behaviors while maintaining realism; if the case is too complicated, it becomes difficult to explain the observed behaviors. The case project proceeded in 5 steps: (1) prefabricated interlocking bricks were transported from the factory to the construction site and stored in the stockyard; (2) one loading robot loaded bricks onto a truck; (3) the truck delivered them from the stockyard to the assembly location within construction site; (4) two assembly robots assembled them to construct a small mock-up bridge; (5) the truck was returned to the stockyard; (Fig. 5). The bridge consisted of a total of 49 bricks of 4 different types (A, B, C, and D) as shown in Fig. 5. In this case project, our testing scope was focused on commercialized industrial robots for easy and intuitive demonstration. Industrial robot may not universally applicable to all construction work, but it can sufficiently demonstrate the necessary task allocation policy in the scenario of this case project. Also, industrial robots were used as an example, but the digital twin framework and DRL algorithm presented in this paper can be applied to any type of construction robot (e.g., transportation robots, inspection robots, and autonomous heavy equipment) for task allocations.

It was assumed that the distance between the stockyard and the bridge site was 100 m, that it takes 10 s to load and unload a brick from the truck, and that the truck can carry up to 4 bricks at a time. When the truck arrives at the stockyard or bridge assembly site, it stops to be loaded and unloaded, and waits or returns (Fig. 5-(1) and Fig. 5-(3)). The bricks required assembly according to the prescribed installation guidelines (e.g., The first brick in the middle line of the bridge should start with brick type B. If a brick that did not meet the guideline was delivered, its assembly could not proceed, and consequently, construction completion was delayed.

5.2. Input modeling to reflect realistic robotic construction challenges

We defined three simulation model parameters to represent dynamic and stochastic construction processes in input modeling process, namely $P1$: brick arrival rate $(number\ of\ arrival/s)$; $\overline{P2}$: average truck speed (m/s); and $P3$: assembly rate $(number\ of\ assembly/s)$. $P1$ and $P3$ are the rate (λ) of the Poisson process, which counts the expected number of events per unit of time. The Poisson process is widely used to represent arrival rate when an event occurs that is independent from preceding events [73–75]. The assembly rate ($P3$) can be treated in the same manner as the arrival rate ($P1$), assuming that new bricks arrive when the previous assembly step is completed. $\overline{P2}$ is the average speed at which the materials handling truck travels between the stockyard and the assembly site. These three parameters represent significant challenges in robotic construction [8,69–72]. Specifically, $P1$ represents a logistics challenge, which changes the availability of bricks at the stockyard. $\overline{P2}$ represents an on-site material transportation challenge and is affected by site layout, the number of workers, and obstacles. $P3$ represents an on-site robotic assembly (e.g., bolt and nut insertion, fixing, and screwing) challenge, which controls the brick assembly rate of two assembly robots. These three parameters can be used to represent a realistic robotic construction environment that includes dynamic and stochastic processes.

Such a realistic environment is necessary to ensure DRL agent training; by manipulating these three parameters, we can generate the input data required to simulate a realistic environment in the digital twin. The above parameters were changed following either Poisson distribution ($P1$ and $P3$) or random distribution ($\overline{P2}$) in preset training ranges ($0.05 < P1 < 0.09\ [1/s]$, $5 < \overline{P2} < 9\ [m/s]$, $0.018 < P3 < 0.032\ [1/s]$), thereby allowing us to train the DRL agent in a more realistic environment. Ideally, the preset interval should be made using real-world datasets such as actual brick arrival rate, allowable truck speed at a real site, and brick assembly rates in practice. Unfortunately, robotic construction has not yet compiled such a universal dataset. Therefore, we had to subjectively set the training ranges, which we did to reflect less idle time for robots (loader robot and assembly robots). It should be noted that there is no claim that such training ranges are the only allowable ranges for this experiment. However, we did set these ranges in a way which we believe optimizes the realism of the simulated environment.

5.3. DRL agent training

The realism of the digital twin environment created through the aforementioned input modeling process better enables us to train a DRL agent to find an adaptive control policy. For each simulation iteration, the DRL agent takes actions which govern each robot according to given site states (i.e., brick availability, truck's status, and assembly status). When directing the loading robot, the agent decides which types of bricks (A to D) and how many (1 to 4) to load on the truck. For the assembly robots, the agent decides in what order to assemble the bricks delivered via the truck and whether to send the truck back to the stockyard or to wait (Fig. 6-(a)); the truck is designed to deliver bricks by

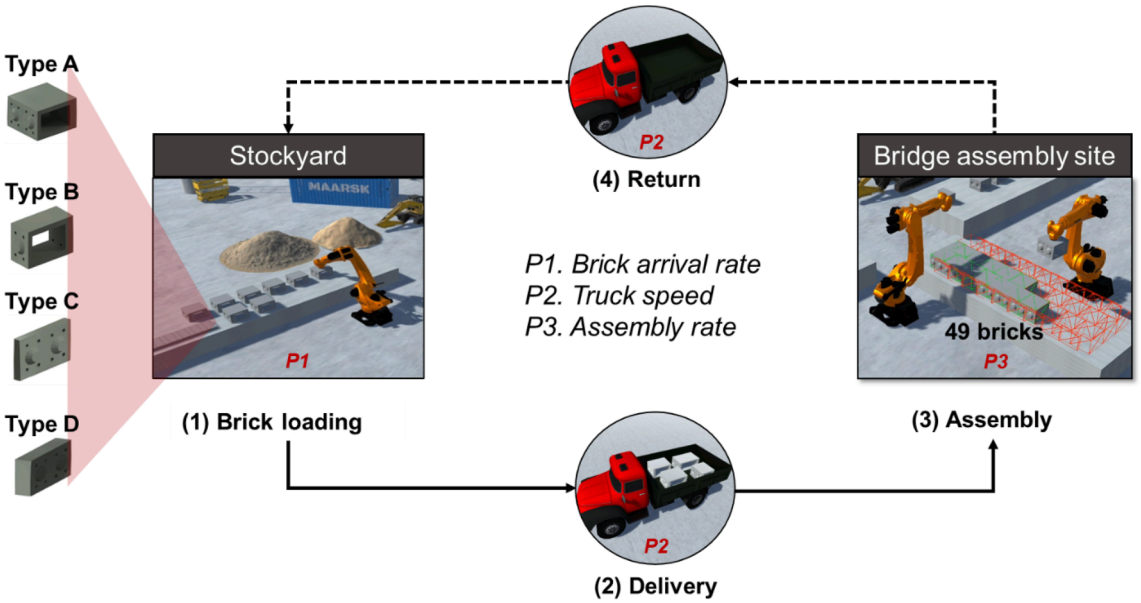


Fig. 5. Case project overview.

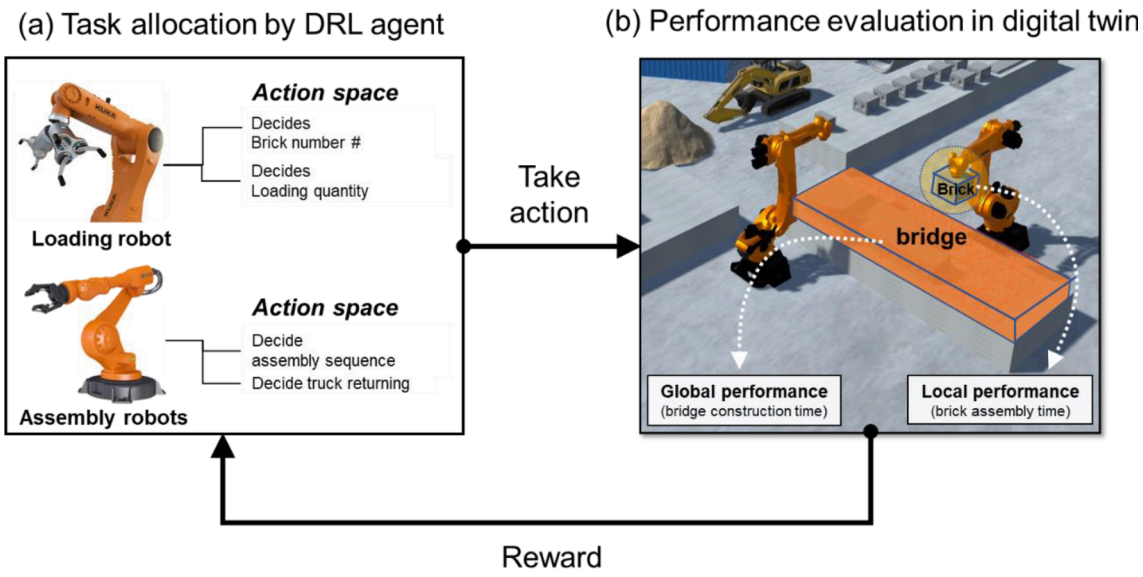


Fig. 6. Interaction flow between DRL agent and digital twin for DRL training.

reciprocating between the stockyard and the bridge assembly site without being controlled by the agent. The DRL agent takes actions for all robots at every iteration, but in the simulation environment, if the truck is near the stockyard, it only activates the loading robot. Likewise, if the truck is near the bridge assembly site the DRL agent only activates assembly robots. A reward for the agent's behavior is returned only based on activated robots' behaviors. Meanwhile, the agent is rewarded for their actions according to the robots' local performance (the time required for a brick assembly) and global project performance (bridge construction time), and this feedback, in turn, updates the policy network (Fig. 6-(b)). The hyperparameters used to train the PPO-based DRL model are shown in Table 1. Hyperparameters were set by referring to values that were optimized and used in previous robotics research in OpenAI [20].

The learning performance of the DRL model was evaluated using two RL indicators: cumulative reward and episode time [76–78]. Cumulative reward per episode, taken over the entire training period, measures whether the reward increases as the episode increases— that is, whether the policy is being updated adaptively for the environment. It can also evaluate whether the plot converges stably. As episode time measures the time it takes to complete an episode (i.e., bridge construction), it measures the global performance of the policy over the entire training period. The shorter the construction time, the higher the performance, so the smaller the episode time, the higher the performance.

The result of the cumulative reward plot fluctuates slightly in the beginning but stabilizes and begins increasing after around 1,000 iterations and then converges after 30,000 iterations (Fig. 7-(a)). This indicates that the DRL agent is not stably adapting to the environment in the beginning, but consistently found a way to increase the reward after 1,000 iterations, and finally found an optimal policy in the environment, which includes dynamic and stochastic conditions, at around 30,000 iterations. On the other hand, episode time per episode decreases after around 1,000 iterations (Fig. 7-(b)), indicating that an increase in reward leads to an increase in performance. Therefore, we may conclude that the policy at 30,000 iterations, where the cumulative reward value is at its maximum, is the optimal policy. If the environment changes, it may be necessary to tune the hyperparameters to find an updated optimal policy. The optimal policy composed of the weight of the policy network can then be extracted and used for testing robot control performance.

Table 1
Hyperparameters for the PPO-based DRL model.

Hyper-Parameter	Value
Learning rate	0.0002
Time horizon	256
Generalized advantage estimator (λ)	0.95
Batch size	256
Update rate (τ)	0.005
Discount factor (γ)	0.9
Buffer size	100,000
Max steps	40,000
Number of hidden layers	2 (128 neurons per layer)
Beta (β)	0.001
Number of epochs	5
Epsilon	0.2

5.4. Adaptivity test of DRL model

We tested the trained DRL model's adaptive control performance in the testing environment. Such an environment displays a wider range of parameter change than those used for the training ranges ($0.01 < P1 < 0.14$ $[1/s]$, $1<\overline{P2} < 14$ $[m/s]$, $0.004 < P3 < 0.05$ $[1/s]$). Evaluating the model trained using preset training ranges for its behavior outside said training ranges would allow us to assess the adaptivity of the trained model to a completely new environment (i.e., never experienced environment while training). Only one parameter was changed at a time, and the remaining parameters were fixed at their median values ($P1$: 0.075 $[1/s]$, $\overline{P2}$: 7.5 $[m/s]$, and $P3$: 0.027 $[1/s]$); this isolated change allowed us to better understand the adaptive behavior for each testing environment. $P1$ and $P3$ changed following Poisson distribution, and $\overline{P2}$ changed following the random distribution. The DRL model, which was trained using a training range, and the rule-based model were each inserted into the aforementioned testing environment to test their behaviors, and their global performance, indicated by total bridge construction time, was compared. A total of three rules are assumed here. First, the loading robot loads bricks onto the truck in the order they first arrived. Second, the truck loads only the maximum quantity of bricks the loading robot can carry at a moment, meaning the truck does not wait until additional bricks arrive. Third, the assembly robots construct the bricks in the order they were delivered in. Analysis demonstrated that the DRL model effectively adapted to three testing environment, showing better performance (i.e., less construction time) compared to the rule-based model (Table 2). Specifically, the results in the training range (i.e., grey shaded area in Fig.8-(a), (b), and (c)) show that the rule-

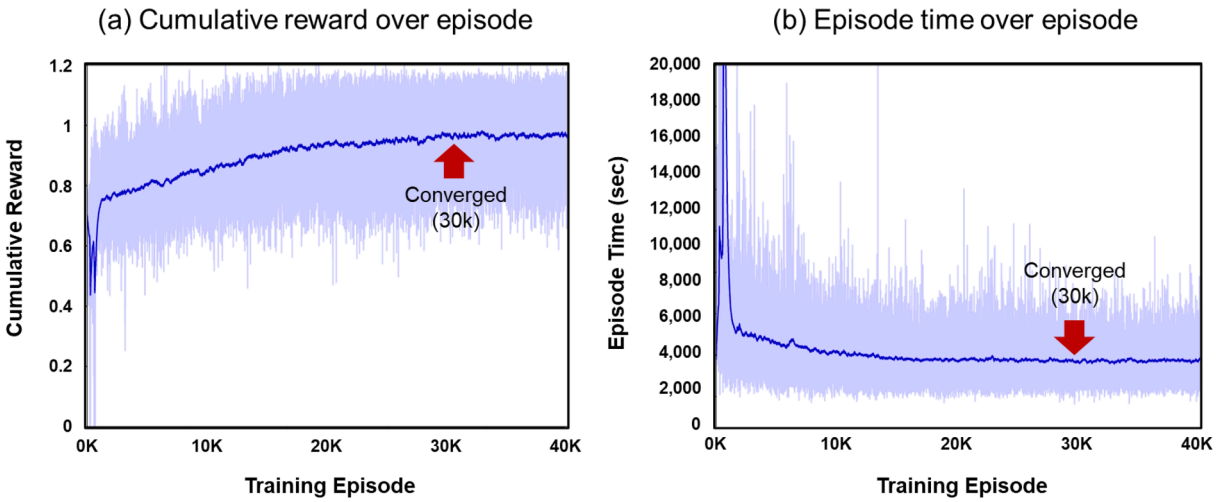


Fig. 7. Learning curve for (a) cumulative reward and (b) episode time.

Table 2
Average construction time (ACT) comparison between Rule-based model and DRL model behaviors by simulation model parameters.

Parameter	ACT in Rule-based model behavior (s)	ACT in DRL model behavior (s)
Brick arrival rate (1/*s*)	6,952	4,071
Truck speed (m /*s*)	5,974	4,100
Assembly rate (1/*s*)	6,832	3,730
Trained range	6,356	3,920
(standard deviation for 840 samples)	(1,346)	(465)
Untrained range	6,685	3,988
(standard deviation for 360 samples)	(1,235)	(713)

based model took an average of about 6,356 s (standard deviation: 1,346 s) to complete the bridge construction, while the DRL model took an average of about 3,920 s (standard deviation: 465) (Fig.8-(d)). The results in the untrained range (i.e., a range other than the training range) show that the rule-based model took an average of about 6,685 s (standard deviation: 1,235 s) to complete the bridge construction, while the PPO-based model took an average of about 3,988 s (standard deviation: 713 s) (Fig.8-(d)). These results indicate that the PPO model showed, overall, 36 % higher average performance (36 % reduced construction time) with 52 % reduced standard deviation (data points are clustered closely around the mean; more reliable), than the results of the rule-based model. Thus, the DRL model is more adaptable than the rule-based model to a robotic construction environment which includes dynamic and stochastic processes. Furthermore, because we found that DRL demonstrates higher performance than the rule-based model, even in new environment profiles with new parameters outside the training range, we may claim a generalization of DRL to an entirely new environment that was not made known to the model during training. In other words, the results confirmed that the agent learned adaptive control policy beyond the specifics of the training range.

Fig. 8 Robot control performance in an environment profile which includes dynamic and stochastic conditions according to changes in (a) brick arrival rate, (b) truck speed, (c) assembly rate, and (d) performance comparison between rule-based model and DRL model both in trained range and untrained range.

6. Discussion

This study developed a digital twin-driven DRL framework to investigate DRL's potential for adaptive task allocation at a robotic construction site. As the experiment demonstrated, the DRL agent can stably find an adaptive policy in a simulated environment and showed promising adaptive task allocation performance (reduced construction time on average by 36 % compared to a rule-based model) in three testing environments which included dynamic and stochastic conditions. One particularly noteworthy finding, which can be seen in the performance comparison plot (Fig. 8-(d)), is that the average construction time decreased consistently across both the trained and untrained ranges when using the DRL model, while exhibiting a smaller value for the standard deviation in the trained range (465 s in training vs 713 s in testing). Here, standard deviation can be an indicator of the reliability of autonomous robot behavior for robots controlled using the DRL model. The discrepancy in standard deviation between the trained and untrained ranges indicates that we should continuously update the training ranges to reflect realistic values so that behaviors of autonomous robots trained in the digital twin can continue adapting reliably even in reality. Since construction environments evolve over time as projects progress, an agent's policy must be constantly updated according to its changing environment. From this point of view, the digital twin has great potential since the digital twin's environment is constantly updated. In such an environment, agents can continuously update their policies through an online learning approach.

The DRL agent was able to find an adaptive task allocation policy in a robotic construction environment which included dynamic and stochastic processes because we assumed that, since the digital twin provided overall site conditions in near real-time, all states in the environment were fully observed. This complete observability helped our DRL agent quickly and stably find an adaptive policy even under dynamic and stochastic conditions. This result indicates that the digital twin can be a core component of DRL learning in environments that include difficult learning conditions (i.e., partially observable space, dynamic and stochastic conditions). With the aid of a digital twin, a robot can take actions toward improving global project performance (e.g., cost, time, quality, safety, and sustainability) by observing the overall states of a construction site rather than relying only on data or states it collects locally. This learning method can be effective in construction projects where global project performance is more important than the local performance of an individual robot.

If robots can adapt to an environment that includes dynamic and stochastic site conditions without specific prior domain knowledge or instructions, they can accomplish a larger set of tasks in more complex and uncertain situations. This adaptivity is particularly important in construction because many different activities are carried out simultaneously, with a high degree of uncertainty inherent in each process with

significant dependencies. If the robots can take the best actions in consideration of dynamic and stochastic processes, they will be able to cope with such uncertainties and consequently improve the overall performance of the project. For example, assembly robots can adaptively change their assembly sequence or find alternative plans when the material delivery status is delayed by an unexpected accident, to avoid any future cost overrun or schedule delay. If a mobile robot predicts a potential safety accident risk, they might proactively change their moving speed and/or route directly to prevent such accidents. In short, robots can autonomously select the best behavioral strategy for successful project execution even in a dynamically changing construction site environment, and ultimately, fully automated construction (LoA 5) may become possible with the adaptivity.

Although this study demonstrated the potential of the suggested framework for adaptive task allocation in the robotic construction environment, there are still several important "scalability" issues to consider. The issue with scale is that as the numbers and types of robots increases, the dimensions of state and action space increase exponentially, and consequently, the DRL agent has trouble finding the optimal policy. This issue is particularly serious in a construction project where a multiplicity of different types of robots conducting various activities must be controlled simultaneously. In this paper, for example, we focused on industrial robot task allocation as a case study but our approach can be applied to any type of robot. If the task of each robot can be parameterized and standardized, any type of task allocation problem can be solved with the same DRL approach. In this situation, we may use a multi-agent system (MAS), comprised of multiple interacting agents. These agents could be software agents, robots, machines, or even humans, and they might have a common goal to solve together and/or each have an individual goal. Each agent has its own policy, which allows more freedom from the scalability issue than does a single-agent system. The introduction of MAS may allow us to find fully autonomous, adaptive, and even collaborative control policies for multiple robots in robotic construction.

Although this study demonstrated the potential of DRL for adaptive robot control, there are several important issues to consider. First, the validation of the proposed framework was performed only in a hypothetical case project. There has as of yet been no implementation of a real-world synchronized digital twin with IoT sensors. Though this hypothetical project serves the objective of this research in testing the potential of digital twin-driven DRL for robot control, it is necessary to validate it with a real-world project in the future in order to confirm real-world applicability. Second, in this study, we selected a construction scenario in which humans and robots did not interact directly. However, if humans are involved or present in which the robots are embedded, various postures and actions must be simulated by modeling a person in a digital twin space. Unlike robots, humans can exhibit various behaviors depending on their mental and physical states. Therefore, detailed human modeling may be required through connection with ergonomics. Third, the DNN model used in this paper is composed of a relatively shallow network with four hidden layers for faster convergence. While learning efficiency increases with shallow networks, they are difficult to apply to complex real-world robot control problems (e.g., robot control in sparse reward, high noise, high dimensions of state and action space, and partially observable environment). Therefore, to improve the generalizability of the model, the architecture should be modified in favor of a slightly deeper network, adjusting to the control problems. Lastly, also, in future research, our DRL based task allocation approach may be linked to well defined or standardized robotic construction processes for automated robot controls. However, each construction project has unique characteristics (e.g., site conditions, construction methods, and tasks) and projects evolve as they progress. Moreover, each construction project may need unique robot controls rather than simple task allocations, and there is no standardized/parameterized robotic construction process or instructions yet, which makes many uncertain situations for a robot. These uncertainties pose severe obstacles to applying DRL in the construction domain. For this reason, even if the digital twin provides a realistic representation in real-time and is used as a learning environment for DRL, there will be a simulation-to-reality gap because, as long as the agent cannot precisely predict the future, the agent always learns a policy from a previous environment. In order to reduce the simulation-to-reality gap and use DRL in practice, model generalization techniques (e.g., domain randomization or domain adaptation methods) should be applied in RL. In addition, the digital twin must be able to simulate diverse scenarios for the generalizability of the RL learning. For example, different worker movement patterns based on historical datasets or future construction processes based on 4D BIM should be created in the digital twin environment that the RL agent interacts with. The RL policy trained in these diverse scenarios may have more generalizability in real-world applications.

7. Conclusion

We developed a digital twin-driven DRL framework to investigate DRL's potential for adaptive task allocation policy formation in robotic construction applications. A digital twin can capture real-world site conditions in near real-time and simulate dynamic and stochastic site conditions for a DRL agent to interact with. The DRL agent is then able to learn an adaptive task allocation policy to increase project performance at a global level given overall state observations. We tested the suggested framework in a case project experiment. The results indicated that the DRL agent can stably find an adaptive policy in a simulated environment. Additionally, we found that the DRL's control behavior resulted in a 36 % construction time decrease compared to the rule-based model in a robotic construction environment that included dynamic and stochastic processes. This finding indicates that the DRL model is more adaptable to an environment than the rule-based model. Therefore, the primary contribution of this paper is to demonstrate the potential of DRL for adaptive task allocation in a robotic construction environment. Such an adaptive allocation method can help robots make optimal decisions through considering past, present, and future site conditions— ultimately enabling robots to cope with uncertainties and to boost project performance.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work presented in this paper was supported financially by MCubed (Project ID 8505 'Digital Twin for Construction and Logistics for Lego-Inspired Construction' and "Digital Lego-inspired Construction"), United States; the Jiangsu Industrial Technology Research Institute (LID:12494320 for support of research on 'Robotic Additive Manufacturing of Smart Construction Blocks'), China, and the National Research Foundation of Korea (NRF) grant funded by the Korea government Ministry of Science and ICT (MSIT) [No. NRF-2020R1A4A4078916 & 2022R1G1A1012897], South Korea. In addition, we wish to thank Wenxuan Zhu, an undergraduate student at the University of Michigan for his assistance in implementing a digital twin and reinforcement learning system.

References

[1] 2019 Worker Shortage Survey Analysis, 2019. https://www.agc.org/sites/default/files/Files/Communications/2019 Worker Shortage Survey Analysis.pdf.
[2] N.V. Schwatka, L.M. Butler, J.R. Rosecrance, An Aging Workforce and Injury in the Construction Industry, Epidemiol. Rev. 34 (2012) 156–167, https://doi.org/10.1093/epirev/mxr020.

[3] F. Barbosa, J. Woetzel, J. Mischke, M. Ribeirinho, M. Sridhar, M. Parsons, N. Bertram, S. Brown, Reinventing Construction: A Route To Higher Productivity, McKinsey & Company. (2017) 12. http://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/reinventing-construction-through-a-productivity-revolution%0Awww.revalue.dk.

[4] D.O. Won, K.R. Müller, S.W. Lee, An adaptive deep reinforcement learning framework enables curling robots with human-like performance in real-world conditions, Sci. Rob. 5 (2020), https://doi.org/10.1126/scirobotics.abb9764.

[5] I. Carlucho, M. De Paula, G.G. Acosta, An adaptive deep reinforcement learning approach for MIMO PID control of mobile robots, ISA Trans. 102 (2020) 280–294, https://doi.org/10.1016/j.isatra.2020.02.017.

[6] National Infrastructure Commission, Data for the public good, National Infrastructure Commission. (2017) 76.

[7] Y. Pan, L. Zhang, A BIM-data mining integrated digital twin framework for advanced project management, Autom. Constr. 124 (2021), https://doi.org/10.1016/j.autcon.2021.103564.

[8] C. Feng, Y. Xiao, A. Willette, W. McGee, V.R. Kamat, Vision guided autonomous robotic assembly and as-built scanning on unstructured construction sites, Autom. Constr. 59 (2015) 128–138, https://doi.org/10.1016/j.autcon.2015.06.002.

[9] F. Tao, Q. Qi, L. Wang, A.Y.C. Nee, Digital Twins and Cyber-Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison, Engineering. 5 (2019) 653–661, https://doi.org/10.1016/j.eng.2019.01.014.

[10] W. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn, Digital Twin in manufacturing: A categorical literature review and classification, IFAC-PapersOnLine. 51 (2018) 1016–1022.

[11] R. Sacks, I. Brilakis, E. Pikas, H.S. Xie, M. Girolami, Construction with digital twin information systems, Data-Centric Eng. 1 (2020).

[12] K.K. Patel, S.M. Patel, Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges, Int. J. Eng. Sci. Comput. 6 (2016).

[13] N. Kousi, C. Gkournelos, S. Aivaliotis, C. Giannoulis, G. Michalos, S. Makris, Digital twin for adaptation of robots' behavior in flexible robotic assembly lines, Procedia Manuf. 28 (2019) 121–126.

[14] S.S. Mousavi, M. Schukat, E. Howley, Deep Reinforcement Learning: An Overview, Lecture Notes in Networks and Systems. 16 (2018) 426–440, https://doi.org/10.1007/978-3-319-56991-8_32.

[15] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT press, 2018.

[16] J. Fu, A. Kumar, O. Nachum, G. Tucker, S. Levine, D4rl: Datasets for deep data-driven reinforcement learning, ArXiv Preprint ArXiv:2004.07219. (2020).

[17] N. Ruiz, S. Schulter, M. Chandraker, Learning to simulate, ArXiv Preprint ArXiv: 1810.02513. (2018).

[18] R. Bellman, A Markovian decision process, J. Math. Mech. 6 (4) (1957) 679–684.

[19] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, PMLR, 2014: pp. 387–395.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms (2017) 1–12.

[21] J. Schulman, P. Moritz, S. Levine, M.I. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, in: 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings, 2016, pp. 1–14.

[22] S. Zhang, Y. Pang, G. Hu, Trajectory-Tracking Control of Robotic System via Proximal Policy Optimization, in: Proceedings of the IEEE 2019 9th International Conference on Cybernetics and Intelligent Systems and Robotics, Automation and Mechatronics, CIS and RAM, 2019. (2019), pp. 380–385, https://doi.org/10.1109/CIS-RAM47153.2019.9095849.

[23] N. Melenbrink, J. Werfel, A. Menges, On-site autonomous construction robots: Towards unsupervised building, Autom. Constr. 119 (2020), 103312, https://doi.org/10.1016/j.autcon.2020.103312.

[24] T. Bock, T. Linner, Robot oriented design, Cambridge University Press, 2015.

[25] T. Bock, T. Linner, Robotic industrialization, Cambridge University Press, 2015.

[26] T. Bock, Construction robotics, Journal of Robotics and Mechatronics. 28 (2) (2016) 116–122.

[27] T. Bock, T. Linner, Site Automation, Cambridge University Press, 2016.

[28] T. Bock, T. Linner, J. Güttler, K. Iturralde, Ambient Integrated Robotics: Automation and Robotic Technologies for Maintenance, Assistance, and Service, Cambridge University Press, Cambridge, 2019. DOI: 10.1017/9781139872034.

[29] M. Gharbia, A. Chang-Richards, Y. Lu, R.Y. Zhong, H. Li, Robotic technologies for on-site building construction: A systematic review, J. Build. Eng. 32 (2020), 101584, https://doi.org/10.1016/j.jobe.2020.101584.

[30] B. Panda, S.C. Paul, L.J. Hui, Y.W.D. Tay, M.J. Tan, Additive manufacturing of geopolymer for sustainable built environment, J. Cleaner Prod. 167 (2017) 281–288, https://doi.org/10.1016/j.jclepro.2017.08.165.

[31] E. Asadi, B. Li, I.M. Chen, Pictobot: A Cooperative Painting Robot for Interior Finishing of Industrial Developments, IEEE Rob. Autom. Mag. 25 (2018) 82–94, https://doi.org/10.1109/MRA.2018.2816972.

[32] S.K. Yun, D. Rus, Adaptive coordinating construction of truss structures using distributed equal-mass partitioning, IEEE Trans. Rob. 30 (2014) 188–202, https://doi.org/10.1109/TRO.2013.2279643.

[33] S. Goessens, C. Mueller, P. Latteur, Feasibility study for drone-based masonry construction of real-scale structures, Autom. Constr. 94 (2018) 458–480, https://doi.org/10.1016/j.autcon.2018.06.015.

[34] P. Latteur, S. Goessens, C. Mueller, Masonry construction with drones, Proceedings of IASS Annual Symposia. 2016 (2016) 1–10. https://www.researchgate.net/publication/316107815_Masonry_construction_with_drones.

[35] K.H. Petersen, R. Nagpal, J.K. Werfel, Termes: An autonomous robotic system for three-dimensional collective construction, Robotics: Science and Systems VII. (2011).

[36] K. Iturralde, M. Feucht, H. Rongbo, W. Pan, M. Schlandt, T. Linner, T. Bock, J.-B. Izard, I. Eskudero, M. Rodriguez, A cable driven parallel robot with a modular end effector for the installation of curtain wall modules, (2020).

[37] Autonomous Haulage System Optimizes Surface Mining | Komatsu America Corp, (n.d.). https://www.komatsuamerica.com/autonomous-haulage-system (accessed October 14, 2020).

[38] Volvo CE unveils the next generation of its electric load carrier concept : Volvo Construction Equipment, (n.d.). https://www.volvoce.com/global/en/news-and-events/press-releases/2017/conexpo-vegas-2017/volvo-ce-unveils-the-next-generation-of-its-electric-load-carrier-concept/ (accessed October 14, 2020).

[39] Brayman Construction and Advanced Robotics Partner to Combat Worker Shortage with TyBot | 2018-08-01 | ACP, (n.d.). https://www.acppubs.com/articles/7483-brayman-construction-and-advanced-robotics-partner-to-combat-worker-shortage-with-tybot.

[40] INTSITE | Powered by AI, (n.d.). http://intsite.ai/.

[41] T. Hwu, A.Y. Wang, N. Oros, J.L. Krichmar, Adaptive robot path planning using a spiking neuron algorithm with axonal delays, IEEE Trans. Cogn. Developm. Syst. 10 (2) (2018) 126–137.

[42] M. Tong, W. Lin, X. Huo, Z. Jin, C. Miao, A model-free fuzzy adaptive trajectory tracking control algorithm based on dynamic surface control, Int. J. Adv. Rob. Syst. 17 (2020).

[43] A. Bakdi, A. Hentout, H. Boutami, A. Maoudj, O. Hachour, B. Bouzouia, Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control, Rob. Auton. Syst. 89 (2017) 95–109.

[44] A.H. Karami, M. Hasanzadeh, An adaptive genetic algorithm for robot motion planning in 2D complex environments, Comput. Electr. Eng. 43 (2015) 317–329.

[45] S. Kundu, D.R. Parhi, Navigation of underwater robot based on dynamically adaptive harmony search algorithm, Memetic Computing. 8 (2) (2016) 125–146.

[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing Atari with Deep Reinforcement Learning (2013) 1–9.

[47] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process Mag. 34 (6) (2017) 26–38.

[48] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533, https://doi.org/10.1038/nature14236.

[49] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search, Nature 529 (2016) 484–489, https://doi.org/10.1038/nature16961.

[50] Y. Ding, L. Ma, J. Ma, M. Suo, L. Tao, Y. Cheng, C. Lu, Intelligent fault diagnosis for rotating machinery using deep Q-network based health state classification: A deep reinforcement learning approach, Adv. Eng. Inf. 42 (2019), 100977, https://doi.org/10.1016/j.aei.2019.100977.

[51] F. Zhang, J. Leitner, M. Milford, B. Upcroft, P. Corke, Towards vision-based deep reinforcement learning for robotic motion control, Australasian Conference on Robotics and Automation, ACRA. (2015).

[52] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, M. Riedmiller, Data-efficient deep reinforcement learning for dexterous manipulation, ArXiv Preprint ArXiv:1704.03073. (2017).

[53] S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, Proceedings - IEEE International Conference on Robotics and Automation. (2017) 3389–3396, https://doi.org/10.1109/ICRA.2017.7989385.

[54] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, Reinforcement and imitation learning for diverse visuomotor skills, ArXiv Preprint ArXiv:1802.09564. (2018).

[55] J.-H. Lee, H.-J. Kim, Reinforcement learning for robotic flow shop scheduling with processing time variations, Int. J. Prod. Res. 1–23 (2021).

[56] F. Bademosi, N. Blinn, R.R.A. Issa, Use of augmented reality technology to enhance comprehension of construction assemblies, ITcon. 24 (2019) 58–79.

[57] S.H. Lee, F. Peña-Mora, M. Park, Dynamic planning and control methodology for strategic and operational construction project management, Autom. Constr. 15 (2006) 84–97, https://doi.org/10.1016/j.autcon.2005.02.008.

[58] J.C. Stegen, X. Lin, A.E. Konopka, J.K. Fredrickson, Stochastic and deterministic assembly processes in subsurface microbial communities, ISME J. 6 (2012) 1653–1664, https://doi.org/10.1038/ismej.2012.22.

[59] R.G. Andrzejak, G. Widman, K. Lehnertz, C. Rieke, P. David, C.E. Elger, The epileptic process as nonlinear deterministic dynamics in a stochastic environment: An evaluation on mesial temporal lobe epilepsy, Epilepsy Res. 44 (2001) 129–140, https://doi.org/10.1016/S0920-1211(01)00195-4.

[60] Y. Zheng, Z. Meng, J. Hao, Z. Zhang, Weighted Double Deep Multiagent Reinforcement Learning in Stochastic Cooperative Environments BT - PRICAI 2018: Trends in Artificial Intelligence, in: X. Geng, B.-.-H. Kang (Eds.), Springer International Publishing, Cham, 2018, pp. 421–429.

[61] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, H. Lee, Sample-efficient reinforcement learning with stochastic ensemble value expansion, in: Advances in Neural Information Processing Systems, 2018: pp. 8224–8234.

[62] M. Roustai, M. Rayati, A. Sheikhi, A. Ranjbar, A scenario-based optimization of Smart Energy Hub operation in a stochastic environment using conditional-value-at-risk, Sustain. Cities Soc. 39 (2018) 309–316.
[63] S.-M. Hung, S.N. Givigi, A Q-learning approach to flocking with UAVs in a stochastic environment, IEEE Trans. Cybern. 47 (1) (2017) 186–197.
[64] G.C. Lopes, M. Ferreira, A. Da Silva Simoes, E.L. Colombini, Intelligent control of a quadrotor with proximal policy optimization reinforcement learning, Proceedings - 15th Latin American Robotics Symposium, in: 6th Brazilian Robotics Symposium and 9th Workshop on Robotics in Education, 2018. (2018), pp. 509–514, https://doi.org/10.1109/LARS/SBR/WRE.2018.00094.
[65] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, P. Abbeel, Adversarial attacks on neural network policies, ArXiv Preprint ArXiv:1702.02284. (2017).
[66] Q. Vuong, Y. Zhang, K.W. Ross, Supervised policy update for deep reinforcement learning, ArXiv Preprint ArXiv:1805.11706. (2018).
[67] P.V. Ingle, G. Mahesh, Construction project performance areas for Indian construction projects, Int. J. Constr. Manage. (2020), https://doi.org/10.1080/15623599.2020.1721177.
[68] A.P.C. Chan, A.P.L. Chan, Key performance indicators for measuring construction success, Benchmarking: An International Journal. (2004).
[69] T. Bonwetsch, Robotically assembled brickwork: Manipulating assembly processes of discrete elements, (2015).
[70] K.H. Petersen, N. Napp, R. Stuart-Smith, D. Rus, M. Kovac, A review of collective robotic construction, Sci. Rob. 4 (28) (2019).
[71] K. Dörfler, T. Sandy, M. Giftthaler, F. Gramazio, M. Kohler, J. Buchli, in: Robotic Fabrication in Architecture, Art and Design 2016, Springer International Publishing, Cham, 2016, pp. 204–217.
[72] L. Ding, W. Jiang, Y. Zhou, C. Zhou, S. Liu, BIM-based task-level planning for robotic brick assembly through image-based 3D modeling, Adv. Eng. Inf. 43 (2020), 100993.
[73] R. Liu, M.E. Kuhl, Y. Liu, J.R. Wilson, Modeling and simulation of nonstationary non-Poisson arrival processes, INFORMS J. Comput. 31 (2) (2019) 347–366.
[74] W. Fischer, K. Meier-Hellstern, The Markov-modulated Poisson process (MMPP) cookbook, Perform. Eval. 18 (2) (1993) 149–171.
[75] R.W. Klein, S.D. Roberts, A time-varying Poisson arrival process generator, Simulation. 43 (4) (1984) 193–195.
[76] F. Pardo, A. Tavakoli, V. Levdik, P. Kormushev, Time limits in reinforcement learning, 35th International Conference on Machine Learning, ICML 2018. 9 (2018) 6443–6452.
[77] P. Tadepalli, D.K. Ok, Model-based average reward reinforcement learning, Artif. Intell. 100 (1998) 177–224, https://doi.org/10.1016/s0004-3702(98)00002-2.
[78] J. Yang, B. Petersen, H. Zha, D. Faissol, Single Episode Policy Transfer in Reinforcement Learning (2019) 1–17.